3 Abstractions and Implementations

An AI must be made concrete and real to do any work in the world.

Unfortunately, at the time of this writing, it has become fashionable to discuss AI in the abstract, as if its mechanisms of action and future behaviors were based on the common experiences we observe in humans and other animals.

There are also those who imagine impossible abstractions that make the most rational choice at every opportunity, or perfectly maximize utility, and then make inferences from this about the future impacts of artificial intelligence. These abstractions are called impossible because such ideals only work out in pure mathematics and are not computable or effectively calculable in any meaningful sense. They provide no *direct* insight into how an *actual* AI implementation operates or might be constructed.

What all these lapses in reasoning have in common is that they are all based on intangible abstractions with no basis in physical reality. They are, in a sense, unreal.

The purpose of this chapter, and indeed the entire Foundations section, is to provide the basic knowledge required to understand why it is important to discuss AI as implementations as opposed to abstractions. Theorizing can be useful, but the danger is in drawing conclusions without basing them in reality. Implementations force the thinker to bring concretion to their ideas. How will this work? What would it look like as a computer program or hardware description? What semantics and patterns would I use as a programmer to develop this? These are some of the questions that should be asked of any source using abstractions to make inferences about the behavior of a system in the absence of well-defined specifications or concrete descriptions.

3.1 Finite Binary Strings

If you can count to one, beginning with zero, you can understand the technical foundations of this book. A set is a collection of things in no particular order. The set

$$\{0, 1\}$$

is the binary alphabet. For comparison, the set for the English alphabet is:

$$\{a, b, c , d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}.$$

Both of these sets are *finite* because they terminate. We are primarily concerned in computing with finite things because reality gives us finite resources to work with. There is finite time to reasonably do something or make a decision; finite space in memory or storage; finite energy to do work, and so forth. All implementations *must* be finite, but may involve infinite *processing*, e.g. an infinite loop involving machine consciousness or a self-update. Being finite places boundaries on what can be realized and meaningfully discussed.

One way or another, making something concrete in software involves an eventual translation to binary. This is not to say that binary has primacy over another representation, but rather, that it is a convenient representation to work with conceptually and shares many *direct* relationships with other areas in computer science and mathematics. Note that this translation to binary still applies even when referring to AI implementations that will be put into custom or configurable hardware, as the logic therein can be duplicated verbatim in software, albeit at potentially significant costs to performance.

Strings are the concatenation of symbols from some alphabet. The sets {1,0} and {0,1} are *identical*, but the strings '10' and '01' are distinct. Quotes are used here to highlight the difference between sets and strings and because this is how they are commonly depicted in many programming and scripting languages. Here is an example of a finite binary string:

'010101010101'.

A computer program is also a finite binary string [1]. As a result of this, every AI implementation can also be interpreted as a binary string. This also applies to organisms [2]. A genome is, in fact, a large string, and admits a binary representation that allows analysis through computational linguistics. This is not to draw any correlation between AI implementations and genetic implementations, as they are in distinct *description languages*, (with a very different execution model) which is the topic of the next section.

With concatenation of strings understood, the Kleene closure [3] naturally follows. Thus, we arrive at the set of all possible finite binary strings,

{0,1}*.

This set itself is *countably infinite* and includes every *finite* combination of 0 and 1, including a special empty set {} representing an empty or null string. {0,1}* is an important set because it provides us with a most fundamental canvas from which we must render any and all AI implementations. It is the *medium* in which AI implementations are instantiated. This is an important distinction: by considering it a medium, we come to understand it as a *space* as opposed to an object or a thing. It is crucial to the understanding of real-time interpretations, as it would not simply skip symbols but create a run using the pattern that represents the absence of something in that description language. For example, consider a hypothetical program storing data from an analog sensor that registers a signal, then drops below the detection threshold, and then rises again:

'11111000011111'.

In many cases, but not all, the absence of something would be a run of 0s. But this is not a rule; in other encoding schemes, the spatial extent of the information is not in correspondence with time. That is, the encoding scheme explicitly has timing and synchronization primitives built into it and is simply atemporal, lacking any notion of time.

This perspective is important because, as a medium, binary is used to embed or *represent* information. It is technically incorrect to say that all information is binary or digital or anything of the kind, as information must be *interpreted* [4, 5]. The symbols signify structure, and that signification can be present in a variety of media, binary being just one of many. Further, there are many ways to encode the same information, and this can also vary by media.

Returning to the Kleene star, it is important to know that the set of all finite binary strings includes all possible AI implementations as just a *subset*. A subset means that there is a set which is "inside" or included within another set, possibly of equal or larger size. The implications being that there will be nonsensical and nonworking programs within {0,1}*. This is because it has all the possible combinations of 1 and 0, including no combinations. The part we are concerned with is the subset that realizes working AI, which will be referred to as the set of all AI implementations. Further, there is another subset within {0,1}* that represents the set of all strong AI implementations.

It is also possible to include the memory, knowledge, and data acquired by a strong AI as part of a definition by concatenating that data to the end of the string, and then defining that as a subset of {0,1}*. This would, however, require a special encoding of the AI implementation so that its length would be included as part of its specification. Each string would represent a totally complete, ready to run implementation until it learns and changes a single bit of information, becoming a new and distinct string. This is a foreshadowing of the chapter on self-modification.

This is a powerful and universal way of analyzing AI implementations. It should be clear, even now, why it is nonsensical to discuss abstractions in the absence of the concrete, well-defined structures of an implementation.

But what is the point of introducing such a low level construct? The primary reason is to provide a basis for discussing practical and concrete implementations. The goal here is to move away from abstractions that are unclear or even impossible. But there is also another reason, and it has to do with the communication of ideas in the field of strong AI, which is a more general statement of the problem this chapter is trying to address.

Most of the sciences have a very specific and complex lexicon to communicate and express their concepts. The science of strong artificial intelligence, as distinct from machine learning and narrow AI, must also have its own methods. Based on what has been shown already, we know that such exchange will be based on at least what can be possibly constructed, and that such implementations need to be specified clearly and concretely.

Mathematics itself might come to mind as a preferred mode of communication for this discipline, but it is not as “natural” a choice as it might first seem. Instead, we should consider programming languages and their related constructs and terminology to provide immediately actionable communication between strong AI scientists. *At the very least*, no major concept should be without the corresponding source code to give concretion to it.

The choice to use programming languages has two justifications: first, it is the ability to run what is given to us without having to translate mathematical symbols and definitions into code that gives it a tremendous benefit, and two, that mathematical concepts have multiple ways to be implemented, creating ambiguity, and leaving much to be desired in terms of actual algorithm implementations. If mathematics could truly replace all our needs to specify, understand, and communicate in computer science, then we wouldn’t have needed to create a separate and distinct field in the first place. Likewise, the needs of strong AI science require clear, rigorous, and unambiguous formal communication for its ideas. Mathematics will be a tool, and, in some cases, a means to prove certain things, but not over and above the programs and algorithms that will ultimately be implementing strong AI and other forms of advanced automation.

3.2 Description Languages

With an understanding of (finite) binary strings, it is now possible to move into description languages and their relationship to AI implementations. The interpretation and use of description languages in AI security is founded on the field of algorithmic information theory [1, 6, 7, 8, 9].

A *description language* is a means of encoding or specifying messages (descriptions). In this context, these descriptions can be referred to as *programs*, with the program being a message in some programming language.

Description languages can be applied to one or more binary strings where there is consistent structure internally or across strings. This could be considered a corpus and corpora, respectively. Machine learning can thus be viewed as a producer of description languages; the description language of a set of one or more messages is modeled, or learned, by exploiting correlations between and within them. The modeled description language is then used to validate, identify, or even generate (predict) messages. However, this is not the complete picture of what is happening with regard to the learned description language. This is because it is possible to recursively encode descriptions so that they become a partial or complete message in some other description language, and that is exactly what is happening in the case of machine learning.

To understand this recursive embedding we will use a simple example: we can specify a description language for PNG files, with the structure of the file format being the description language and the pixel data and other information being the description. Such a PNG could then be encapsulated or embedded into a ZIP archive, with that format’s structure as its description language, and its descriptions being inclusive of messages whose description languages are foreign to it. Since ZIP programs are able to treat their archived data as opaque, they don’t require knowledge of the description language to work with them. This is true even of the compression that is used on the archived data, which relies on analysis of patterns within the data without having explicit knowledge of their format.

This leads to the more complex case with machine learning. For example, an artificial neural network could be considered a description language and its weights and training information its descriptions. But it goes at least one level deeper. The descriptions could be interpreted as models, which would have description languages themselves. And it is the fidelity of a model that determines its predictive (generative) power. This results in at least two nested levels of description languages, not including the programming language and machine level implementation of the artificial neural network. Any machine learning algorithm can be substituted in the above example, as each must make some kind of model or representation of something in order to identify and predict it.

By understanding this recursive property of description languages we gain the ability to universally analyze implementations, algorithms, and data structures across domains. This does not just apply to computer science, but to any information which has a consistent structure; it could be chemical compounds, enzymatic reactions, or genomes. It could be recipes or instructions on how to build something. All of these can be viewed as messages in one or more description languages, any of which can be interpreted as finite binary strings. This gives us measurable and objective facts to work with that allow analysis of complexity, integrity, and other useful properties. Because they are concrete, we would have the ability to perform tests and experiments, and reason about their exact behavior. This is not possible with simple discussions in the abstract, as we may interpret them in different ways, leading ultimately to differing implementations, or the abstraction may in fact be intractable or impossible to construct.

3.3 Conceptual Baggage

First, it must be pointed out that the field of narrow AI and strong AI are to be considered distinct. That is one of the minor themes of this book, which is essential to understanding the security and safety challenges. This is true even when discussing AI as a whole as both narrow and strong AI systems belong to this category.

The foundation of this problem is that the field of strong AI lacks an identity as a scientific discipline. Its appears to have boundaries which are in flux or non-existent. Authority figures from philosophy, computer science, mathematics, and even physics and the biological sciences flood the conceptual space with a barrage of ideas and prediction, most of it being nonsensical or lacking any concretion. This ordinarily wouldn't be a negative, as this is just humanity trying to grapple with a difficult concept, but this field has a very different set of circumstances surrounding it that must be acknowledged.

Strong AI has a *massive* set of cultural and psychological attachments that go along with it. This *conceptual baggage* retards growth and makes for an almost impossible atmosphere for education. It is a state of intellectual chaos, with the default being that anyone is qualified to discuss it because we're all supposedly experts on intelligence being intelligent beings ourselves; the more intelligent society thinks a person is, the more we accept that they are qualified to discuss the nature of intelligence. This problem is enabled by an anthropocentric bias and driven by the psychological need for social signaling in intellectual circles.

It would be absurd to trust a physicist to do neurosurgery based on the argument that both physics and neurosurgery were both intellectually challenging, or, that because all brains are ultimately made of the same physical stuff as the rest of the universe that the physicist was somehow qualified. No rational person would let this argument justify allowing the physicist to perform their surgery. Yet, moral differences notwithstanding, this is *exactly* what is happening in the field of artificial intelligence, and it's not just physicists but people from many different fields, backgrounds, and levels of expertise.

There is a psychological gap that is not being minded, an anthropocentric blind-spot, hidden by the shared mutual drive to speak about intellect as the apogee of social affluence. This exists in both speakers and those that promote them, less these waves of sensationalism would have dissipated instantly. But they continue to be propagated and promulgated because we're not collectively rejecting the source as abstract and vague. We have not built up an intellectual immunity to these ideas because we are primed to admit them, at least tacitly, due to the closeness of the subject with our own nature.

Concepts and abstractions must be constructed in order for them to do work in the world. Further, every concept that can potentially do work has *at least* one implementation. Natural language is inherently prone to misunderstanding and misinterpretation, and this is worsened by the fact that not every abstraction that seems reasonable has an *effective*, let alone efficient, implementation. These problems can be eliminated by recognizing that the conventions we use for describing or discussing AI behavior, especially with respect to safety and security, *must be reducible to some finite binary string as a description in some machine language or its equivalent*. This forces us to stop and question ourselves as to whether what we are discussing or reading makes any sense.

One of the the most popular examples of this conceptual baggage is to interpret AI as rational agents and then predict their behavior using utility functions and decision theory, backed up by probability and statistics [10, 11, 12, 13, 14, 15, 16, 17, 18, 19].

The problem with this concept is that an AI implementation is physically vulnerable to failure and attack [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33], which is a practical issue that agent concepts, utility functions, and the decision-theoretic are fundamentally incapable of addressing. Rationality only makes sense given first a set of background assumptions about the values and goals that define what it considers to be sensible decisions and actions. Without this, it can not be applied. Further, no finite set of values can be used to entail all possible AI implementations, no matter how reasonable they seem. More to the point, even if these values could be entailed, their encoding would be just as vulnerable as the AI implementation itself, even if designed into the architecture itself.

Finally, and most importantly, the background assumptions and values for what one defines as rational do not constitute an actual model of the behavior of the AI implementation. That is to say, a model of consequences as a function of value(s) can not be accurate without the nuances of implementation details. This is especially true of an instance of strong AI. These models will also fail to address contextual ramifications or unanticipated outcomes. It lacks the ability to determine *how* these environments and situations are interpreted, as it doesn't have knowledge of the inner workings of the implementation. In other words, these models assume a perfect implementation that can never be realized. This is because there will eventually arise situations were judgments are compromised due to interference or miscalculation. Failure of the AI implementation to arrive at the expected outcome will always be skewed by a varying and unknowable amount of ambiguity and error in any given context. Such challenges require modeling that has full knowledge of the relevant implementation details. Anything less is incoherent.

To reiterate: even if the architecture was implemented in a decision-theoretic framework, it would still not eliminate the physical and contextual ambiguity of a perception pipeline, among other factors in the implementation, nor would it remove the inherent non-zero probability of error that exists in these implementations. Further, no amount of architectural inclusion or closeness to this conceptual baggage will eliminate its physical vulnerability as information in software or hardware; it is ultimately a red-herring in the search for practical solutions to the safety and security of artificial intelligence. As a result of this, it should rightly be considered an approach that is non-workable or non-applicable to the challenges of AI safety and security.

3.4 Anthropocentric Bias

The focus of this section is to refute the tendency to believe that AI functionality and behavior can be predicted by extrapolating and applying human behavior, either derived scientifically, or, more commonly, through folk-psychological [34, 35, 36] accounts.

This bias is damaging to the security and safety of AI because of its limiting effect on the mind to assess the vulnerabilities and operation of AI implementations; it creates a mismatch between what an AI implementation *will* do and what one *believes* it will do.

A consequence of this bias, in conjunction with the conceptual baggage surrounding this field, has created a belief that the immediate threat to humanity is from advanced artificial intelligence itself [37, 38, 39], when, in fact, it will be people utilizing this technology for malicious purposes that will present the threat. The media then repeats this misinformation and it gets disseminated to the public, countermanding efforts at public outreach and education on these issues. As a result, these biases are setting us back in a very real way, and we will continue to be unprepared as long as the focus is fixed on moral intelligence and the delusion of a singular, personified strong AI arising out of all possible AI implementations to subvert the human race.

There is no law of nature that states that an AI must be implemented based on the human condition. More generally, there is no law of nature that an AI be restricted to biologically inspired designs. The burden of proof is on those who believe that out of all the possible AI implementations in {0,1}* that each must be based on our limited cognitive framework. It's trivial to show that it's possible to construct programs that are *nothing* like biology, let alone how our brains work, yet are capable of accomplishing similar tasks. The following is a complete program that counts from 1 to 10:

```
for i in range(10): print i + 1
```

Just as previously introduced, the above program is a *description* in a *description language* called Python, a general purpose programming language. The human brain is nothing like this description, both in terms of how it accomplishes it and in terms of simplicity. It took billions of years of evolution to enable the human brain to have the capacity to learn to do what this program does in just one line of code. Both can count to 10 and both are reasonably effective at doing this task.

Possible counter-arguments to this simple evidence might be that it doesn't represent an actual AI implementation; that it's too simple. However, it's purpose is to show that it is possible to automate the process of counting from 1 to 10. That it's not an artificial neural network or based on millions of n-grams from a corpus of numerical sequences and counting systems is irrelevant. Further, the description to have a neural network duplicate this program's external behavior would not only be incomprehensible and opaque to us, but it would require a vastly larger number of steps to simulate on a digital computer. One could try to argue that this could be accelerated by specialized hardware for simulating that neural network ontology, but that could be countered by the fact that an integrated circuit designed to do this digitally would be a fraction of the complexity. That the above program is incapable of adapting or learning new number systems is also irrelevant to showing the effective equivalence in the tasks.

The point is that there are descriptions that will yield similar or equivalent results that need not be based on identical or approximated biological descriptions; a simulation or modeling of our biological functioning is not a necessary condition to realize equivalent results. Any counter to this point would have to explain why it is impossible to effectively calculate some processes and not others. That is to say, what special properties of the world are off limits to the information-theoretic, and why? Any answer to this will have to overcome the overwhelming observational accounts of the Church-Turing thesis applying to physical systems [40] in everyday use, and the continual pushing back of what we thought wasn't possible for computing and automation to achieve.

A related bias is to assume that digital computers will never be conscious because they are not made of the same substance as human brains [50, 41, 42], or, that there is something unique to either biological or non-biological neuronal processing [43], and that, as a result, AI implementations not based on this will never achieve the same level of functioning. However, that we lack a rigorous 1st person account of conscious experience [44] is not evidence against machine consciousness nor does it imply that we must turn to biological mimicry or exotic metaphysical accounts of mind to achieve it. It simply means we have yet to uncover its description.

3.5 Existential Primer

The most universal way to clear out the misconceptions surrounding artificial intelligence is to start at the very bottom. Within {0,1}* there are no concepts such as agency, ego, or emotion. We will not find consciousness, qualia, or experiences. It is an existentially bleak landscape, a blank canvas from which to draw upon. There is only, at best, a sequence that can be interpreted and computed to realize one or more processes.

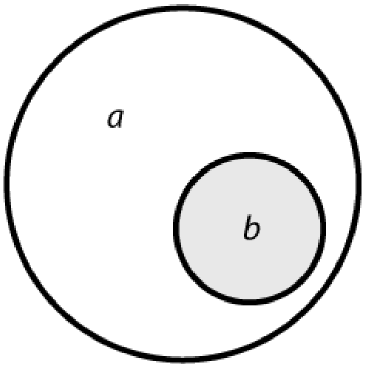
These processes may give rise to some of the aforementioned things when executed, but this does not constitute their existence. A description of a thing makes an abstraction real only insofar as tangibly entailing its potential. That is, with debt to Whitehead's original process philosophy [45], I claim a distinct interpretation that bridges the gulf between his metaphysics and algorithms: descriptions are static representations of time-like objects which can only be realized through one or more processes. To understand, consider the shadows from geometry that arise from the projection of a higher-dimensional object onto a lowerdimensional space. Likewise, the static descriptions that entail processes are but a shadow of their full time-like extents. One could create an enriched static description of such a process through a non-deterministic representation that includes every possible state of the object at every infinitesimal moment in time. Such a representation could also be made through the creation of a uniform stochastic model that treats all events as equally likely.

However, even with such an enriched description we would not have *realized* the thing which it entails. Even fully specified (non-deterministic) static descriptions of processes must be interpreted to become manifest. To do so, it must undergo information exchange, which is, in a more general sense, computation. This must not be confused or conflated with information exchange or complexity as consciousness [46, 47, 48], which is certainly false and outside the scope of this discussion to refute. To address it in short: information exchange is necessary but not sufficient for consciousness. This is also why computationalism is false. This is related to the concept of strong AI that was defined by John Searle in [49], in which he used the term to refute the computational theory of mind. Strong AI used today, as defined in the Introduction of this book, turns Searle's argument on its head and requires that strong AI have the necessary constructs that would give rise to the processes involved in consciousness. That is to say, strong AI must be a cognitive architecture.

Definition: A *cognitive architecture* is a constructable implementation design with features that will allow it to *understand*, have *mental content*, and undergo conscious *experience*.

Recall the set of all possible AI implementations, which is a subset of the set of all finite binary strings. There exists another subset of the set of all finite binary strings that is of interest. It is the set of all possible cognitive architectures:

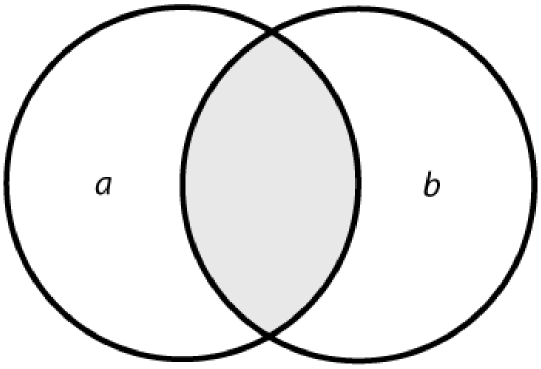
Figure 3.1: The set of cognitive architectures as a proper subset of the set of all finite binary strings.



Where ***a*** is the set of all finite binary strings and ***b*** is the set of all cognitive architectures.

The set of cognitive architectures and the set of AI implementations are not identical, but they do have an intersection:

Figure 3.2: Intersection between the set of AI implementations and the set of cognitive architectures.



Where ***a*** is the set of all AI implementations and ***b*** is the set of all cognitive architectures.

It is at this intersection that we describe any possible notion of conscious processing or machine understanding. It is within this intersection that strong AI will have to be constructed. This is not something that can be realized accidentally, but must be forged through engineering. There are a potentially infinite number of alternative AI implementations that do not yield a cognitive architecture in any meaningful sense. As a result, it should be considered nonsensical to impute agency, consciousness, or any other properties or features that are not present in the description of the AI implementation.

To refer to AI categorically as a collective, "species", or group, is to commit to error. There is no law of nature that AI implementations share a common link, identity, or connection. This is because each AI implementation will be a unique instance, with potentially distinct features, knowledge, and information making up its construction. Moreover, it will have a unique vantage point, given that it occupies a distinct position in time and space. As a result of this, it will necessarily have a unique frame of reference. AI implementations will require network and communications features to overcome this default state of physical independence and individuation. It is a complex engineering task that will not arise spontaneously without an effective process that yields it. This also applies to the extended case of a single cognitive collective or unified mind across multiple physical entities or instances. This falls under the set of all cognitive architectures mentioned above.

Lastly, AI implementations will *not* have the tendency to convergence towards a singular entity or diverge from a unified identity to multiple individual ones. This will not occur without internal mechanisms and design features or the necessary external and environmental pressures to guide self-modification. That is to say, it is incoherent to assume that AI implementations—of any level of intelligence—will work either for or against this type of (self-)organization. Nor can any *general* argument be made for or against this case, as it would require a specific context, set of background assumptions, and a precise description of the AI implementation.

3.6 AI Implementations

Everything so far in this chapter has been leading up to a discussion of AI implementations. This term has been used several times in advance of its definition in order to establish a context and to set it apart from the relevant issues. It has been shown that abstractions and simple discussion can not account for the operational level details nor the complexity of AI implementations, and that they must be used in order to avoid conceptual baggage and anthropocentrism. The existential and ontological assumptions have been addressed and hopefully wiped away. The canvas should now be clear. All that remains is to more clearly define this term and to briefly cover some of its details at a high-level.

Definition: An *AI implementation* is a valid and working description of an artificial intelligence, of any level of complexity, that may either be interpreted or executed on a computer or equivalently translated hardware specification.

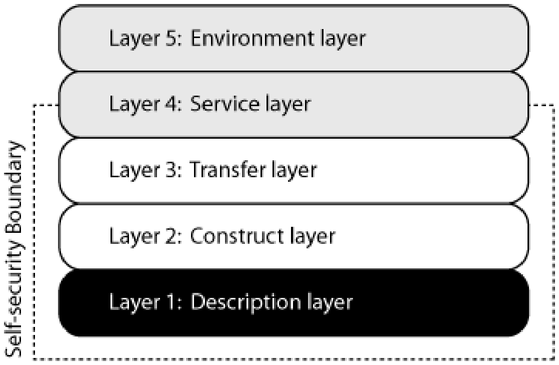
Recall that the set of all AI implementations is a subset of the set of all finite binary strings. This means that any specification and design for an AI implementation, as per this definition, *must* be constructable as a description in some description language. This description does not have to be worked with in binary. For example, it could be presented as a programming language, which might be translated to a hardware description or register transfer language. It could be a set of very detailed programming schematics at the semantic level of the language itself. Ultimately, it is implied that any description language can be equivalently encoded in binary or is represented that way as part of the natural operation of the system’s information processing and storage, even when the description itself is being presented to the user in a human-readable representation.

There is no one set of architectural rules or laws that make up an AI implementation, and, as previously discussed, it is not possible to generalize all possible AI implementations and reason from them in the abstract.

One of the critical perspectives of this book is that there can be no assurances of the safety of an AI implementation without understanding and analyzing its security. All the safeguards and moral intelligence in idealized perfection are meaningless if compromised or circumvented. And, given that all forms of self-security can ultimately be overcome with effort, the focus must be on mitigation with the assumption of failure. From this, we analyze the situation that AI implementations will be used in, and look for commonalities in both the environment and the implementations themselves. Analysis of behavior, outcomes, and vulnerabilities must be analyzed uniquely at every stage of design and implementation, with expertise and understanding of *how* a concept or abstraction is translated into a construct at each relevant stage.

The security analysis of an AI implementation begins with the **AIS** model. In almost all cases, if one lower-level layer is compromised then all subsequent higher layers will be compromised:

Figure 3.3: The **AIS** model is comprised of multiple cascading layers of dependent security.



A compromise, fault, or failure at any level causes the layers above it to also become compromised. Self-security entails any and all internal mechanisms for the security and safety of the implementation and must not be exclusively relied upon. The environment layer has the final responsibility in ensuring the containment and behavioral constraints of AI implementations.

Layer 1: Description layer. A universal layer concerned with vulnerabilities that arise from intentional and unintentional modification of the description itself. The primary concern here is with integrity and authentication of the descriptions; mitigating modification and detecting if and when it has occurred.

Layer 2: Construct layer. This layer is what the description entails or constructs. It is directly dependent upon the idioms, semantics, and syntax of the description language. This will usually be a programming language. Faults on this layer may result in working implementations that, nonetheless, have improper and vulnerable descriptions due to bugs and design flaws. Expert knowledge is required in order to translate from concepts to specifications and finally to semantics, especially with security in mind.

Layer 3: Transfer layer. This would be whatever the final description would be to allow the implementation to be interpreted, executed, or operated. This may be the same as layer 2 if the description language is an interpreted language. The purpose of this layer is to focus on post-translation properties and descriptions, which could have been the result of multiple-stages of processing and intermediate representation(s). The primary concern is with the output description that will be sent to the target platform, be it real, emulated, or virtual. In the case of hardware description languages, this would be the final description before being physically instantiated as hardware components and interconnects. Failures at this layer could stem from incorrect compilation or constructs in optimization and compilation that create vulnerabilities in the low-level description. Examples might include the use of dynamically linked or shared libraries; debugging information not being stripped; monolithic, non-compartmentalized execution, and so forth, among many others.

Layer 4: Service layer. This layer is focused on the underlying machine, interpreter, and/or hardware being used to run the AI implementation. In the **AIS** model the boundary for self-security goes through it. This implies that this layer is capable of breaking the pure self-security limitation through physically distributed designs and those which can reasonably be designated as being separated from a single underlying model of execution. Failures at this layer could result in transient soft-errors or faults in hardware that cause data loss and corruption. Physical damage and tampering may interfere with previous stages of security by directly circumventing or manipulating the way in which the system processes and updates the implementation. Safeguards at this layer would include physical security measures, tamper resistance and detection, among other methods.

Layer 5: Environment layer. This level is concerned with everything external to the AI implementation itself. This focuses on an analysis of the hazards relevant to the physical deployment and use of the AI and the risks it presents to life and property. Security on this level would involve traditional methods of physical security, along with additional safeguards in the event of failure or breach in containment. This is perhaps the most important layer as it represents a last line of defense in the event that an implementation ceases to operate under safe, expected behavior. It is also the first layer that is completely independent from the self-security of the underlying impelementation, and should be much more difficult to overcome, with multiple fail-overs. Lastly, this layer applies to a broad range of contexts where confinement is geographic in scope. The same principles apply.

A detailed discussion of these layers and how they can be instrumented to make AI implementations more robust and secure is given later in the book. They are only sketched here so that we may move quickly through the fundamentals. These are prerequisites for the prerequisite of understanding why this type of security is so important, which ultimately comes from the unavoidable future where people have global access to unrestricted strong AI.

The purpose of this chapter has been to establish the need to address the way we communicate and discuss artificial intelligence, especially with regard to their future impacts on humanity. A focus on AI implementations, backed up by an understanding of description languages, solves this challenge and will allow forward progress on the issues that transcends the ambiguities of natural language. This also paves the way for a foundation that is aligned with the fact that AI security presupposes AI safety. This is something that the pure abstraction and hand-waiving can not address, and is, in fact, in stark contrast to the current mainstream understanding. Believing that moral intelligence is the road to AI safety is to engage in a dangerous line of reasoning that is not substantiated by the science. As was shown in the **AIS** model, it only takes a failure at one layer for the rest of the system to become compromised. And all forms of moral intelligence and rule-following for human values will always be at a more dependent layer than that of the description, which can be modified, shared, and distributed freely among people once they have extracted or leaked these copies.

If nothing else is remembered from this chapter, let at least this fact remain vivid: *moral intelligence, rule-following, and all internal safeguards we put into or around an AI implementation are forms of self-security and are presupposed by real world problems and challenges.* No amount of abstract hand-waiving or intellectualizing over the future impacts makes a difference if we do not realize the fact that AI security *presupposes* AI safety. To even begin to address these challenges requires a change in communication from the conceptual to the concrete. We must eliminate our preconceived anthropocentric notions and biases, block out conceptual baggage, and demand discussion at the implementation level of detail. Descriptions are a rigorous and well-defined concept that admit connections to wide range of sub-fields within computer science and mathematics. The practical everyday use need not be concerned with binary strings, but in the textual representation of formal systems like programming languages and equivalent specifications. Anything less than this should be considered nonsensical. There is no simple way to convey or predict the nuances of these systems without an analysis at each stage of the implementation.

References

1. G. J. Chaitin, “A theory of program size formally identical to information theory,” *Journal of the ACM (JACM)*, vol. 22, no. 3, pp. 329–340, 1975.
2. D. B. Searls, “The computational linguistics of biological sequences,” *Artificial intelligence and molecular biology*, vol. 2, pp. 47–120, 1993.
3. J. R. Shoenfield, *Mathematical logic*, vol. 21. Addison-Wesley Reading, 1967.
4. C. S. Peirce, “Logic as semiotic: The theory of signs,” 1902.
5. J. Lacan, “The mirror stage,” *New Left Review*, vol. 51, pp. 71–77, 1968.
6. R. J. Solomonoff, “A formal theory of inductive inference. Part I,” *Information and control*, vol. 7, no. 1, pp. 1–22, 1964.
7. C. S. Wallace and D. L. Dowe, “Minimum message length and Kolmogorov complexity,” *The Computer Journal*, vol. 42, no. 4, pp. 270–283, 1999.
8. A. N. Kolmogorov, “Three approaches to the quantitative definition of information*,” *International Journal of Computer Mathematics*, vol. 2, no. 1–4, pp. 157–168, 1968.
9. G. Chaitin, “The limits of reason,” *Scientific American*, vol. 294, no. 3, pp. 74–81, 2006.
10. S. Legg and M. Hutter, “Universal intelligence: A definition of machine intelligence,” *Minds and Machines*, vol. 17, no. 4, pp. 391–444, 2007.
11. J. Schmidhuber, “Optimal ordered problem solver,” *Machine Learning*, vol. 54, no. 3, pp. 211–254, 2004.
12. P. Turrini, D. Grossi, J. Broersen, and J.-J. C. Meyer, “Forbidding undesirable agreements: a dependence-based approach to the regulation of multi-agent systems,” in *Deontic Logic in Computer Science*, Springer, 2010, pp. 306–322.
13. C. Castelfranchi, “Modelling social action for AI agents,” *Artificial Intelligence*, vol. 103, no. 1, pp. 157–182, 1998.
14. C. Pearce, B. Meadows, P. Langley, and M. Barley, “Social planning: Achieving goals by altering others’ mental states,” in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence. Quebec City, Canada: AAAI Press*, 2014.
15. E. J. Horvitz, J. S. Breese, and M. Henrion, “Decision theory in expert systems and artificial intelligence,” *International journal of approximate reasoning*, vol. 2, no. 3, pp. 247–302, 1988.
16. J. A. Feldman and Y. Yakimovsky, “Decision theory and artificial intelligence: I. A semantics-based region analyzer,” *Artificial Intelligence*, vol. 5, no. 4, pp. 349–371, 1975.
17. J. A. Feldman and R. F. Sproull, “Decision theory and artificial intelligence ii: the hungry monkey*,” *Cognitive Science*, vol. 1, no. 2, pp. 158–192, 1977.
18. C. P. Langlotz, L. M. Fagan, S. W. Tu, B. I. Sikic, and E. H. Shortliffe, “A therapy planning architecture that combines decision theory and artificial intelligence techniques,” *Computers and Biomedical Research*, vol. 20, no. 3, pp. 279–303, 1987.
19. M. P. Wellman, “Fundamental concepts of qualitative probabilistic networks,” *Artificial Intelligence*, vol. 44, no. 3, pp. 257–303, 1990.

20. I. V. Krsul, “Software vulnerability analysis,” Purdue University, 1998.

21. G. McGraw, *Software security: building security in*, vol. 1. Addison-Wesley Professional, 2006.

22. A. Arora, R. Telang, and H. Xu, “Optimal policy for software vulnerability disclosure,” *Management Science*, vol. 54, no. 4, pp. 642–656, 2008.

23. S. Al-Fedaghi, “System-based approach to software vulnerability,” in *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, 2010, pp. 1072–1079.

24. B. Liu, L. Shi, Z. Cai, and M. Li, “Software vulnerability discovery techniques: A survey,” in *Multimedia Information Networking and Security (MINES), 2012 Fourth International Conference on*, 2012, pp. 152–156.

25. P. Li and B. Cui, “A comparative study on software vulnerability static analysis techniques and tools,” in *Information Theory and Information Security (ICITIS), 2010 IEEE International Conference on*, 2010, pp. 521–524.

26. F. Wotawa, “On the relationship between model-based debugging and program slicing,” *Artificial Intelligence*, vol. 135, no. 1, pp. 125–143, 2002.

27. L. Burnell and E. Horvitz, “Structure and chance: melding logic and probability for software debugging,” *Communications of the ACM*, vol. 38, no. 3, p. 31–ff, 1995.

28. T. A. Cargill and B. N. Locanthi, “Cheap hardware support for software debugging and profiling,” *ACM SIGARCH Computer Architecture News*, vol. 15, no. 5, pp. 82–83, 1987.

29. C. Zamfir and G. Candea, “Execution synthesis: a technique for automated software debugging,” in *Proceedings of the 5th European conference on Computer systems*, 2010, pp. 321–334.

30. R. L. Glass, “Real-time: The ‘lost world’ of software debugging and testing,” *Communications of the ACM*, vol. 23, no. 5, pp. 264–271, 1980.

31. B. Hailpern and P. Santhanam, “Software debugging, testing, and verification,” *IBM Systems Journal*, vol. 41, no. 1, pp. 4–12, 2002.

32. S. M. Srinivasan, S. Kandula, C. R. Andrews, and Y. Zhou, “Flashback: A lightweight extension for rollback and deterministic replay for software debugging,” in *USENIX Annual Technical Conference, General Track*, 2004, pp. 29–44.

33. T. Cipresso and M. Stamp, “Software reverse engineering,” in *Handbook of Information and Communication Security*, Springer, 2010, pp. 659–696.

34. S. P. Stich, *From folk psychology to cognitive science: The case against belief*. the MIT press, 1983.

35. M. Davies and T. Stone, “Folk psychology: The theory of mind debate,” 1995.

36. R. M. Gordon, “Folk psychology as simulation,” *Mind & Language*, vol. 1, no. 2, pp. 158–171, 1986.

37. N. Bostrom, *Superintelligence: Paths, dangers, strategies*. Oxford University Press, 2014.

38. E. Yudkowsky, “Artificial intelligence as a positive and negative factor in global risk,” *Global catastrophic risks*, vol. 1, p. 303, 2008.

39. L. Muehlhauser and A. Salamon, “Intelligence explosion: Evidence and

import," in *Singularity Hypotheses*, Springer, 2012, pp. 15–42.

40. D. Deutsch, "Quantum theory, the Church-Turing principle and the universal quantum computer," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 1985, vol. 400, pp. 97–117.
41. G. Strawson, "Realistic monism: Why physicalism entails panpsychism," *Journal of consciousness studies*, vol. 13, no. 10/11, p. 3, 2006.
42. A. Revonsuo, *Consciousness: the science of subjectivity*. Psychology Press, 2009.
43. P. M. Churchland and P. S. Churchland, "Could a. Machine Think?," *Machine Intelligence: Perspectives on the Computational Model*, vol. 1, p. 102, 1998.
44. J. Levine, "Materialism and qualia: The explanatory gap," *Pacific philosophical quarterly*, vol. 64, no. 4, pp. 354–361, 1983.
45. A. N. Whitehead, *Process and reality*. Simon and Schuster, 2010.
46. D. Balduzzi and G. Tononi, "Qualia: the geometry of integrated information," *PLoS computational biology*, vol. 5, no. 8, p. e1000462, 2009.
47. G. Tononi, *Phi: A Voyage from the Brain to the Soul*. Pantheon Books, 2012.
48. C. Koch, *The quest for consciousness*. New York, 2004.
49. J. R. Searle, "Is the brain's mind a computer program," *Scientific American*, vol. 262, no. 1, pp. 26–31, 1990.
50. G. Strawson and others, "Consciousness and its place in nature," Charlottesville, VA: Imprint Academic, 2006.

▲ Return to Top